Enclosure C: Clean Version of Entire Set of Pending Claims

Application No: 09/336,990

Filing Date: 06/21/1999

Applicant: Jia Xu

Title: A Method of Scheduling Executions of Periodic or Asynchronous Real-Time Processes Having Hard or Soft Deadlines

Art Unit: 2156

Examiner: Kenneth Tang

Mailed: January 23, 2003

At: Toronto, Canada

58. A method of scheduling on one or more processors, executions of a plurality of processes, comprising:

(A)

automatically generating a pre-run-time schedule comprising mapping from a set of periodic process executions to a sequence of time slots on one or more processor time axes, each of the time slots having a beginning time and an end time, reserving each one of the time slots for execution of one of the periodic processes, the positions of the end time and the beginning time of each of the time slots being such that execution of the periodic processes, including satisfaction of predetermined constraints and relations comprising worst-case computation time, period, deadline, permitted range of offset constraints, and exclusion relations can be completed between the beginning time and end time of respective time slots,

(B)

during run-time using the information in the pre-run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to schedule the process executions.

59. A method as defined in claim 58, including the step of converting at least one asynchronous process to a corresponding new periodic process prior to the mapping step, and mapping the new periodic process in a manner similar to mapping of other periodic processes.

60. A method as defined in claim 58, including the step of converting one or more asynchronous processes into corresponding new periodic processes prior to the mapping step, and mapping new periodic processes in a manner similar to mapping of other periodic processes, such that predetermined constraints for said being converted asynchronous processes comprising worst-case computation time, deadline, and minimum time between two consecutive requests constraints, will be satisfied.

61. A method as defined in claim 58, including further executing a set of non-converted asynchronous processes during run-time of the processor at times which do not interfere with execution of processes contained in the pre-run-time schedule.

62. A method as defined in claim 58 including, following pre-run-time scheduling and during run-time of the processor, the step of scheduling executions of a specified set of pe-

riodic and asynchronous processes such that all known ones of said specified constraints and relations will be satisfied, the specified constraints and relations further comprising, for asynchronous processes, worst-case computation time, deadline, minimum time between two consecutive requests, and beginning time and end time of every time slot reserved for every periodic process execution in the pre-run-time schedule.

63. A method as defined in claim 58 including, following pre-run-time scheduling and during run-time of the processor, the step of scheduling executions of a specified set of periodic and asynchronous processes such that all known ones of said specified constraints and relations will be satisfied, the specified constraints and relations further comprising, for asynchronous processes, worst-case computation time, deadline, minimum time between two consecutive requests, and beginning time and end time of every time slot reserved for every periodic process execution in the pre-run-time schedule.

64. A method as defined in claim 58, including scheduling, within the pre-run-time schedule, a difference between the end time and the beginning time of each of said periodic time slots with sufficient time capacity for execution of asynchronous processes that have less latitude than considered ones of periodic processes in meeting their respective deadlines.

65. A method as defined in claim 60 including scheduling, within the pre-run-time schedule, a difference between the end time and the beginning time of each of said periodic time slots with sufficient time capacity for execution of unconverted asynchronous processes that have less latitude than considered ones of periodic processes in meeting their respective deadlines.

66. A method for automatically adjusting lengths of periods of a predetermined set of periodic processes, comprising generating a set of reference periods, setting the length of the period of each periodic process to the length of the largest reference period that is no larger than an original period of the periodic process to form adjusted periods, and storing the adjusted periods for subsequent use in scheduling executions of the periodic processes.

67. A method as defined in claim 58, including prior to the mapping step, automatically adjusting lengths of periods of a predetermined set of periodic processes, generating a set of

reference periods, setting the length of the period of each periodic process to the length of the largest reference period that is no larger than an original period of the periodic process to form adjusted periods, and storing the adjusted periods for subsequent use in pre-run-time scheduling of executions of the periodic processes.

68. A method as defined in claim 65, including prior to the mapping step, automatically adjusting lengths of periods of a predetermined set of periodic processes by generating a list of reference periods, setting the length of the period of each periodic process to the length of the largest reference period that is no larger than an original period of the periodic process to form adjusted periods, and storing the adjusted periods for subsequent use in pre-run-time scheduling of executions of the periodic processes.

69. A method as defined in claim 66, including setting the length of each reference period to be equal to the product of n powers, the base of each of the n powers being a distinct prime number, the n bases of the n powers being the first and smallest n prime numbers, the exponent of each of the n powers being bounded by an exponent upperbound.

70. A method as defined in claim 69, including controlling tradeoff between differences between original period lengths and the adjusted period lengths, and the length of a least common multiple of the adjusted period lengths, by values of the n exponent upperbounds and the value of n.

71. A method of determining whether an asynchronous process should or should not be converted into a new periodic process, by calculating whether a ratio of processing capacity of the processor which is required to be reserved for the new periodic process, to a processor capacity that is required for the asynchronous process if left unconverted, exceeds a predetermined threshold value.

72. A method of converting a set of asynchronous processes having a worst-case computation time, minimum time between two requests characteristics and deadline constraints, into a set of new periodic processes having worst-case computation time, period, deadline, and permitted range of offset constraints, and reducing possible timing conflicts with other periodic or asynchronous processes with less latitude in meeting their deadlines, by taking into

consideration the computation time requirements of the latter processes when determining the deadline of each of the new periodic processes.

73. A method of converting a set of asynchronous processes having a worst-case computation time, minimum time between two requests characteristics and deadline constraints, into a set of new periodic processes having worst-case computation time, period, deadline, and permitted range of offset constraints, wherein a permitted range of offset of each new periodic process being is a subinterval of an interval or a full interval that begins at the earliest time that the corresponding being converted asynchronous process can make a request for execution, and ends at a time equal to the sum of the earliest time that said being converted asynchronous process can make a request for execution plus the period length of the new periodic process minus one time unit.

74. A method as defined in claim 73, including reducing possible timing conflicts with other periodic or asynchronous processes with less latitude in meeting their deadlines, by taking into consideration the computation time requirements of the latter processes when determining the deadline of the new periodic process.

75. A method as defined in claim 58 including, prior to generating the pre-run-time schedule, determining whether each asynchronous process should or should not be converted into a new periodic process, converting a subset of a predetermined set of asynchronous processes having worst-case computation time, minimum time between two requests characteristics and deadline constraints, which have been determined to be convertible, into a set of new periodic processes having worst- case computation time, period, deadline, and permitted range of offset constraints, and reducing possible timing conflicts with other periodic or asynchronous processes with less latitude in meeting their deadlines, by taking into consideration the computation time requirements of the latter processes when determining the deadline of the new periodic process.

76. A method as defined in claim 58 including, prior to generating the pre-run-time schedule, determining whether each asynchronous process should or should not be converted into a new periodic process, converting a subset of a predetermined set of asynchronous processes having a worst-case computation time, minimum time between two requests characteristics

and deadline constraints which have been determined to be convertible, into a set of new periodic processes having release time, worst- case computation time, period, deadline, and permitted range of offset constraints, wherein a permitted range of offset of each new periodic process being is a subinterval of an interval or a full interval that begins at the earliest time that the corresponding being converted asynchronous process can make a request for execution, and ends at a time equal to the sum of the earliest time that said being converted asynchronous process can make a request for execution plus the period length of the new periodic process minus one time unit.

77. A method as defined in claim 75, in which the determining step is performed by calculating whether a ratio of processing capacity of the processor which is required to be reserved for new periodic processes, to processor capacity that is required for the asynchronous process if left unconverted, exceeds a predetermined threshold value.

78. A method as defined in claim 58, including generating the pre-run-time schedule as a feasible two-part pre-run-time-schedule for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run- time schedule.

79. A method as defined in claim 58, including generating the pre-run-time schedule as a feasible two- part pre-run-time schedule on a plurality of processors for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run- time schedule.

80. A method as defined in claim 58, further including the steps of generating the pre-run-schedule by constructing a schedule for executions of the periodic processes within an interval starting from zero and having length equal to maximum offset value plus a bounded number of times of the length of a least common multiple of the periodic process periods, conditions for determining feasibility requiring the existence of a point in the pre-run-time schedule wherein starting from the latter point the schedule repeats in subschedule interval lengths equal to a least common multiple of lengths of the periodic process periods, timing of all executions of all periodic processes within a time interval having length equal to the length of the least common multiple of the periodic process periods being included in each said repeating subschedule interval, and including satisfaction of all predetermined constraints and relations for all executions of all periodic processes within the subschedule interval starting from time zero and ending at said point plus the length of the least common multiple of the periodic process periods in the schedule.

81. A method as defined in claim 80, including generating said pre-run-time schedule on a plurality of processors.

82. A method as defined in claim 58, further comprising the steps of:

(a) generating feasible said pre-run-time schedule for the execution of a set of hard deadline periodic processes, and of a set of soft deadline periodic processes,

(b) assigning a criticality level and a deadline upper-limit to each soft deadline periodic process,

(c) in the case of not finding a feasible pre-run- time schedule under said constraints and relations, identifying a soft critical set that contains soft deadline periodic processes for which

modifying the deadlines of one or more processes in said soft critical set is necessary to meet the deadlines of all hard deadline processes,

(d) repeatedly selecting one process with a lowest criticality level among processes for which its criticality level has not reached the deadline upper- limit in the soft critical set in each case in which a feasible pre-run-time schedule has not been found, increasing the deadline of the selected process by an amount that does not exceed the deadline upper-limit of the selected process and repeatedly attempting to find a feasible pre-run-time schedule until either a feasible pre-run-time schedule is found or all the deadline upper-limits of processes in the soft critical set have been reached without finding a feasible schedule, and indicating the event of either the inability of finding a feasible schedule or of finding a feasible pre-run- time schedule, and indicating the critical set when unable to find a feasible schedule,

(e) recomputing worst-case response times of all hard deadline asynchronous processes after a feasible pre-run-time schedule has been found for all hard and soft deadline periodic processes, and in every event that the worst-case response time exceeds the deadline of a hard deadline asynchronous process, repeatedly selecting one process that has a lowest criticality level among all soft deadline periodic processes that contribute to the worst-case response time of the asynchronous process and increasing the deadline of the selected soft deadline periodic process until either (i) the worst-case response time of every hard deadline asynchronous process is less than or equal to its deadline or (ii) all the deadline upper limits of the particular set of soft deadline periodic processes that contribute to the worst-case response time of some hard deadline asynchronous process that exceeds the deadline of said asynchronous process have been reached, and indicating the event of the latter events (i) or (ii) and the particular set.

83. A method as defined in claim 58, further comprising:

(a) during run-time, or during a pre-run-time phase, using the information in the pre-run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to determine, for any point in time, whether there exists a possibility that immediate execution of a particular asynchronous process may cause the execution of any periodic process with less latitude in meeting a deadline of the latter periodic process as compared with latitude of meeting the deadline of said asynchronous process, to be delayed beyond a predetermined time limit, even if the periodic process is not ready for execution at

said any point in time, and

(b) during run-time, delaying execution of said asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any point in time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any point in time.

84. A method as defined in claim 58, further comprising:

(a) either during run-time, or during a pre-run-time phase, using the information in the pre-run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to determine, for any point in time, whether there exists a possibility that immediate execution of a particular asynchronous process may cause the execution of any periodic process with less latitude in meeting a deadline of the latter periodic process as compared with latitude of meeting the deadline of said asynchronous process, to be delayed beyond the end of the time slot of the periodic process in the pre-run-time schedule, even if the periodic process is not ready for execution at said any point in time, and

(b) during run-time, delaying execution of said asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any point in time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any point in time.

85. A method as defined in claim 58, further comprising:

(a) either during run-time, or during a pre-run-time phase, using the information in the pre-run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to determine, for any point in time, whether there exists a possibility that immediate execution of a particular asynchronous process may cause the execution of said asynchronous process, or the execution of some other asynchronous process, to extend beyond the beginning of the time slot of any periodic process that has not yet started in the pre-run-time schedule, even if the periodic process is not ready for execution at said any point in time, and

(b) during run-time, delaying execution of said asynchronous process if said possibility is

found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any point in time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any point in time.

86. A method as defined in claim 83, including carrying out the method on a plurality of processers.

87. A method as defined in claim 84, including carrying out the method on a plurality of processers.

88. A method as defined in claim 85, including carrying out the method on a plurality of processers.

89. A method as defined in claim 58, further comprising:
(a) either during run-time, or during a pre-run-time phase, using the information in the pre-run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to determine, for any point in time, whether there exists a possibility that immediate execution of a particular asynchronous process may cause the execution of any periodic process to be delayed beyond the end of the time slot of that periodic process in the pre-run-time schedule, even if the periodic process is not ready for execution at said any point in time, and
(b) during run-time, delaying execution of said asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any point in time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any point in time.

90. A method as defined in claim 89, including carrying out the method on a plurality of processers.

91. A method as defined in claim 58, further comprising:
(a) either during run-time, or during a pre-run-time phase, using the information in the pre-

run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to determine, for any point in time, whether there exists a possibility that immediate execution of a particular first asynchronous process may cause execution of any second asynchronous process to be continuously blocked for a duration of the execution of the first asynchronous process and a duration of the execution of any periodic process, when the second asynchronous process has less latitude in meeting the deadline of the second asynchronous process as compared with the latitude of both the first asynchronous process in meeting the deadline of the first asynchronous process and the latitude of the periodic process in meeting the deadline of the periodic process, even if neither the periodic process nor the second asynchronous process are ready for execution at said any point in time, and

(b) during run-time, delaying execution of the first asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of the first asynchronous process at said point in time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any point in time.

92. A method as defined in claim 91 including carrying out the method on a plurality of processers.

93. A method as defined in claim 58, comprising during run-time, detecting, in a case in which no asynchronous process or periodic process that has started is to be immediately put into execution, conditions of whether there exists an execution of some first periodic process that is ready for execution and has not completed execution, and there does not exist any other execution of some second periodic process that has not yet completed, such that execution of the second periodic process is ordered before execution of the first periodic process in the pre-run-time schedule, and the time slot of the first periodic process is not nested within the time slot of the second periodic process in the pre-run-time schedule, and there does not exist any other execution of some third periodic process that is ready and has not completed execution, such that execution of the third periodic process is nested within the time slot of the first periodic process in the pre- run-time schedule, and beginning execution of the first periodic process immediately in the event said conditions are true.

94. A method as defined in claim 58, including determining a worst-case response time of an asynchronous process that has not been converted into a periodic process using a formula

comprising:

the sum of worst-case computation times of asynchronous processes and periodic processes that have less or equal latitude as compared with the latitude of the asynchronous process in meeting their respective deadlines,

plus the maximum time that the asynchronous process may possibly be blocked by some asynchronous or periodic process that has greater latitude as compared with the latitude of the asynchronous process in meeting their respective deadlines,

plus the worst-case computation time of the asynchronous process multiplied by the number of periodic processes with which the asynchronous process has an exclusion relation.

95. A method as defined in claim 58, including, during a pre-run-time phase, determining by simulation a worst-case response time of an asynchronous process corresponding to a feasible pre-run-time schedule of periodic processes consisting of an initial part of the pre-run-time schedule which may be of zero length and a repeating part of the pre-run-time schedule, wherein for each point in time from zero to the end time of the repeating part of the run-time schedule minus one time unit, simulating execution of the asynchronous process using functions used to determine a run-time schedule and recording response time of the asynchronous process under the assumption that the asynchronous process arrives at a point in time under consideration, all other asynchronous processes that can possibly block the asynchronous process arriving at one time unit prior to the point in time under consideration, and all asynchronous processes that have less latitude than latitude of the asynchronous process arriving at the same said point of time under consideration, scheduling executions of periodic processes to start at the beginning time and to complete execution at the end time of their respective time slots in the pre-run-time schedule, wherein whenever the asynchronous process is delayed because it may block execution of some periodic process having less latitude than the latitude of the asynchronous process, or may block execution of some second asynchronous process for the duration of more than one execution of processes having greater latitude as compared with the latitude of the second asynchronous process, all asynchronous processes having less latitude as compared with the latitude of the asynchronous process is delayed in order to delay the asynchronous process for a maximum possible amount of time, thus simulating all possible worst-case scenarios of executions of the asynchronous process.

96. A method as defined in claim 58, including restricting every periodic process in the pre-run-time schedule to be executed strictly within its time slot.

97. A method as defined in claim 58, including, during a pre-run-time phase, generating tables of safe start time intervals for the executions of asynchronous processes, wherein every periodic process in the pre-run-time schedule is scheduled to be executed strictly within its time slot, wherein for every point in time of the pre-run-time schedule, it is determined whether each asynchronous process should be delayed, under the assumption that the actual start time of execution of every periodic process is equal to the beginning time of its time slot, and the actual end time of execution of every periodic process is equal to the end time of its time slot, wherein for every point in time of the pre-run-time schedule, in the event said asynchronous process is to be delayed according to the assumptions, the point in time is set to be unsafe and recorded in a corresponding entry in the table for the point in time and said asynchronous process.

98. A method as defined in claim 58, including, during a pre-run-time phase, generating tables of safe start time intervals for the executions of asynchronous processes, wherein every periodic process is scheduled to be executed strictly within its time slot in the pre-run-time schedule, wherein for selected points in time of the pre-run-time schedule, it is determined whether each asynchronous process should be delayed, under the assumption that the actual start time of execution of every periodic process is equal to the beginning time of its time slot, and the actual end time of execution of every periodic process is equal to the end time of its time slot, wherein for selected points in time of the pre-run-time schedule, in the event said asynchronous process is to be delayed according to the assumptions, that point in time is set to be unsafe and recorded in a corresponding entry in the table for the point in time and said asynchronous process.

99. A method as defined in claim 58 comprising during run-time, detecting at least one event of, at any point in time, whether some asynchronous process has arrived by said point in time, whether some asynchronous process or periodic process has completed its computation at said point in time, and whether said point in time is both the release time and beginning time of a time slot in the pre-run-time schedule for some periodic process, and activating a run-time scheduler at said point in time, and should said at least one event have occurred, de-

termining whether any asynchronous process that has arrived but has not yet been completed should be delayed or immediately put into execution, and including the further step of delaying execution of a first asynchronous process when execution of some other asynchronous process that excludes a periodic process with less or equal latitude as compared with the latitude of the first asynchronous process in meeting their respective deadlines has already started but has not yet been completed.

100. A method as defined in claim 58 comprising during run-time, detecting at least one event of, at any point in time, whether some asynchronous process has arrived by said point in time, whether some asynchronous process or periodic process has completed its computation at said point in time, and whether said point in time is both the release time and beginning time of a time slot in the pre-run-time schedule for some periodic process, and activating a run-time scheduler at said point in time, and should said at least one event have occurred, determining whether any asynchronous process that has arrived but has not yet been completed should be delayed or immediately put into execution, and including the further step of delaying execution of an asynchronous process in the event there exists the possibility that immediate execution of the latter asynchronous process may cause execution of a first periodic process with less or equal latitude to be delayed, when execution of the first periodic process may be preempted by execution of some second periodic process, and the latter asynchronous process cannot be preempted by the second periodic process.

101. A method as defined in claim 58 comprising during run-time, detecting at least one event of, at any point in time, whether some asynchronous process has arrived by said point in time, whether some asynchronous process or periodic process has completed its computation at said point in time, and whether said point in time is both the release time and beginning time of a time slot in the pre-run-time schedule for some periodic process, and activating a run-time scheduler at said point in time, and should said at least one event have occurred, determining whether any asynchronous process that has arrived but has not yet been completed should be delayed or immediately put into execution, and including the further step of delaying execution of an asynchronous process when it is not allowed to preempt execution of any first process that excludes some other second asynchronous process which has latitude that is less than both said any first process and the latitude of said second asynchronous pro-

cess, whereby blocking of the second asynchronous process by the duration of more than one execution of processes with greater latitude thereof may be avoided.

102. A method as defined in claim 58 comprising during run-time, detecting at least one event of, at any point in time, whether some asynchronous process has arrived by said point in time, whether some asynchronous process or periodic process has completed its computation at said point in time, and whether said point in time is both the release time and beginning time of a time slot in the pre-run-time schedule for some periodic process, and activating a run-time scheduler at said point in time, and should said at least one event have occurred, determining whether any asynchronous process that has arrived but has not yet been completed should be delayed or immediately put into execution, and including the further step of delaying execution of an asynchronous process to allow preemption of execution of the asynchronous process by execution of a first periodic process that has latitude less than or equal to the latitude of the asynchronous process, when the asynchronous process does not exclude the first periodic process and does not exclude any other asynchronous process with a latitude that is less than the latitude of the first periodic process, and there does not exist execution of some second periodic process that has not been completed such that execution of the second periodic process is ordered before execution of the first periodic process and execution of the time slot of the first periodic process is not nested within the time slot of the second periodic process in the pre-run-time schedule.

103. A method as defined in claim 58, including, during a pre-run-time phase, determining by simulation a worst-case response time of an asynchronous process corresponding to a feasible pre- run-time schedule of periodic processes , wherein for each point in time from zero to the end time of the repeating part of the run-time schedule minus one time unit, simulating execution of said asynchronous process using functions used to determine a run-time schedule and recording response time of said asynchronous process under the assumption that said asynchronous process arrives at a point in time under consideration, all other asynchronous processes that can possibly block said asynchronous process arriving at one time unit prior to the point in time under consideration, and all asynchronous processes that have less latitude than latitude of said asynchronous process arriving at the same said point of time under consideration, scheduling executions of periodic processes to start at the beginning time and to

complete execution at the end time of their respective time slots in the pre-run-time schedule, simulating all possible worst-case scenarios of executions of said asynchronous process.

104. A method as defined in claim 58, wherein said predetermined constraints and relations further comprise precedence relations.

105. A method as defined in claim 104, including the step of converting one or more asynchronous processes into corresponding new periodic processes prior to the mapping step, and mapping new periodic processes in a manner similar to mapping of other periodic processes, such that predetermined constraints for said being converted asynchronous processes comprising worst-case computation time, deadline, and minimum time between two consecutive requests constraints, will be satisfied.

106. A method as defined in claim 104, including scheduling, within the pre-run-time schedule, a difference between the end time and the beginning time of each of said periodic time slots with sufficient time capacity for execution of asynchronous processes that have less latitude than considered ones of periodic processes in meeting their respective deadlines.

107. A method as defined in claim 104 including, prior to generating the pre-run-time schedule, determining whether each asynchronous process should or should not be converted into a new periodic process, converting a subset of a predetermined set of asynchronous processes having worst-case computation time, minimum time between two requests characteristics and deadline constraints, which have been determined to be convertible, into a set of new periodic processes having worst- case computation time, period, deadline, and permitted range of offset constraints, and reducing possible timing conflicts with other periodic or asynchronous processes with less latitude in meeting their deadlines, by taking into consideration the computation time requirements of the latter processes when determining the deadline of the new periodic process.

108. A method as defined in claim 107, in which the determining step is performed by calculating whether a ratio of processing capacity of the processor which is required to be reserved for new periodic processes, to processor capacity that is required for the asynchronous process if left unconverted, exceeds a predetermined threshold value.

109. A method as defined in claim 104 including, prior to generating the pre-run-time schedule, determining whether each asynchronous process should or should not be converted into a new periodic process, converting a subset of a predetermined set of asynchronous processes having a worst-case computation time, minimum time between two requests characteristics and deadline constraints which have been determined to be convertible, into a set of new periodic processes having release time, worst- case computation time, period, deadline, and permitted range of offset constraints, wherein a permitted range of offset of each new periodic process being is a subinterval of an interval or a full interval that begins at the earliest time that the corresponding being converted asynchronous process can make a request for execution, and ends at a time equal to the sum of the earliest time that said being converted asynchronous process can make a request for execution plus the period length of the new periodic process minus one time unit.

110. A method as defined in claim 104, including generating the pre-run-time schedule as a feasible two-part pre-run-time-schedule for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run- time schedule.

111. A method as defined in claim 104, further including the steps of generating the pre-run-schedule by constructing a schedule for executions of the periodic processes within an interval starting from zero and having length equal to maximum offset value plus a bounded number of times of the length of a least common multiple of the periodic process periods,

conditions for determining feasibility requiring the existence of a point in the pre-run-time schedule wherein starting from the latter point the schedule repeats in subschedule interval lengths equal to a least common multiple of lengths of the periodic process periods, timing of all executions of all periodic processes within a time interval having length equal to the length of the least common multiple of the periodic process periods being included in each said repeating subschedule interval, and including satisfaction of all predetermined constraints and relations for all executions of all periodic processes within the subschedule interval starting from time zero and ending at said point plus the length of the least common multiple of the periodic process periods in the schedule.

112. A method of scheduling on one or more processors, executions of a plurality of processes, comprising:

(A)

automatically generating a pre-run-time schedule comprising mapping from a specified set of periodic process executions to a sequence of time slots on one or more processor time axes, each of the time slots having a beginning time and an end time, reserving each one of the time slots for execution of one of the periodic processes, the positions of the end time and the beginning time of each of the time slots being such that execution of the periodic processes, including satisfaction of predetermined constraints and relations comprising worst-case computation time, period, deadline, and precedence and exclusion relations can be completed between the beginning time and end time of respective time slots,

a difference between the end time and the beginning time of each of the time slots with sufficient time capacity for execution of asynchronous processes that have less latitude than considered ones of periodic processes in meeting their respective deadlines,

(B)

during run-time, using the information in the pre-run-time schedule, including the positions of the beginning time and end time of the time slots of the periodic processes, to make scheduling decisions for execution of the processes.

113. A method as defined in claim 112 including, prior to the mapping step, converting one or more asynchronous processes having a worst-case computation time, minimum time between two requests characteristics and deadline constraints, into a set of new periodic pro-

cesses, and reducing possible timing conflicts with other periodic or asynchronous processes with less latitude in meeting their deadlines, by taking into consideration the computation time requirements of the latter processes when determining the deadline of each of the new periodic processes, and mapping the new periodic process in a manner similar to mapping of other periodic processes.

114. A method as defined in claim 113, in which the determining step is performed by calculating whether a ratio of processing capacity of the processor which is required to be reserved for new periodic processes, to processor capacity that is required for the asynchronous process if left unconverted, exceeds a predetermined threshold value.